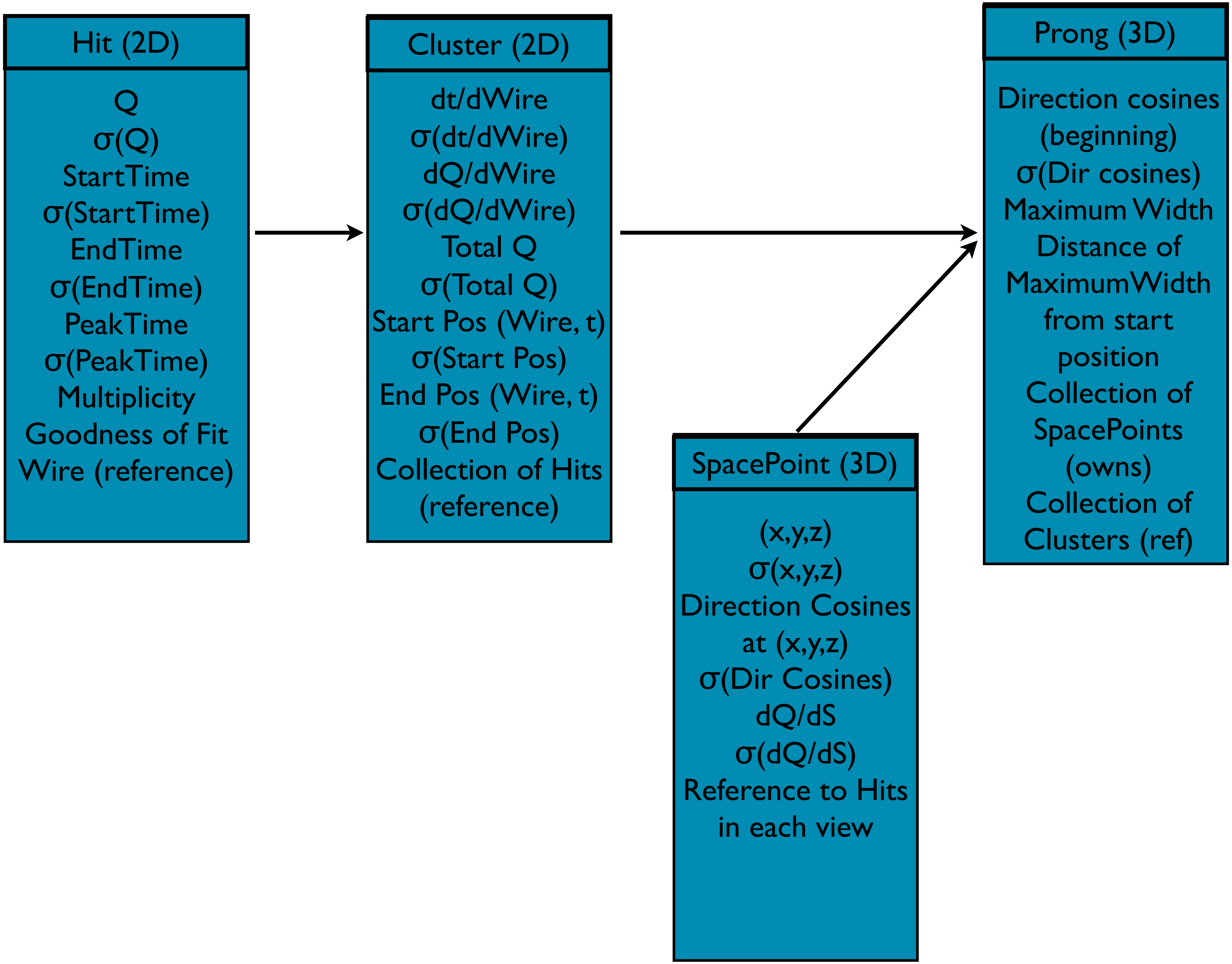
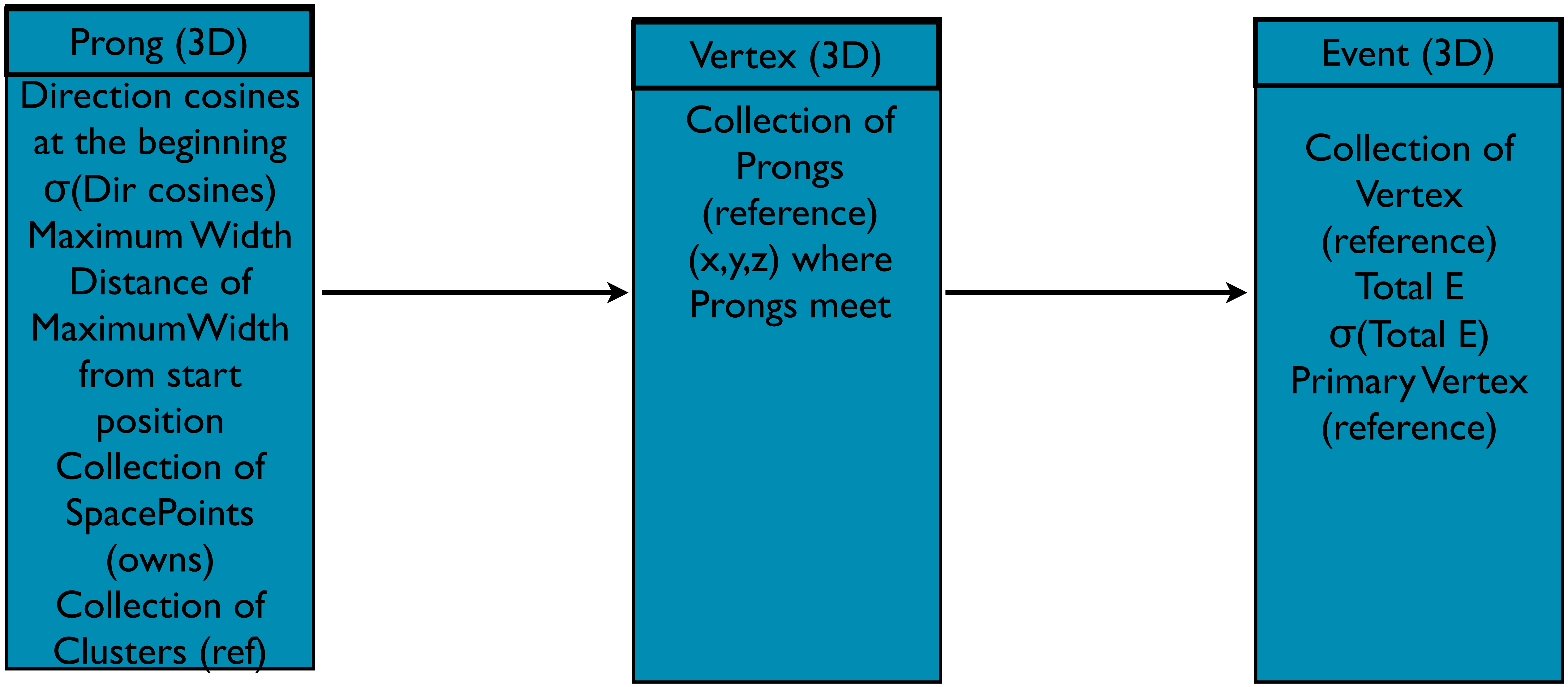
Proposed Reorganization of Reconstruction Base Classes

Brian Rebel

Motivation

- The current organization doesn't appear to be tenable to finish the reconstruction chain
  - There is inherent confusion as to the role of prongs and clusters because they can currently be either 2D or 3D
  - We don't have a very good way to make use of the information gained using several of the current and innovative algorithms that have been developed - too many algorithms are producing basically the same kind of objects that then need to be combined to produce a more complete example of exactly the same type of object
  - We won't get a clear 3D reconstruction without some reorganization
  - We need additional objects in order to do proper analyses
- The next 2 slides show the objects and their data members for the new organization, methods are not shown
- Distinctions are made as to which objects are to be 2D and which are 3D


Hit (2D)
Q
σ(Q)
StartTime
σ(StartTime)
EndTime
σ(EndTime)
PeakTime
σ(PeakTime)
Multiplicity
Goodness of Fit
Wire (reference)
Cluster (2D)
dt/dWire
σ(dt/dWire)
dQ/dWire
σ(dQ/dWire)
Total Q
σ(Total Q)
Start Pos (Wire, t)
σ(Start Pos)
End Pos (Wire, t)
σ(End Pos)
Collection of Hits
(reference)
Prong (3D)
Direction cosines
(beginning)
σ(Dir cosines)
Maximum Width
Distance of
MaximumWidth
from start
position
Collection of
SpacePoints
(owns)
Collection of
Clusters (ref)
SpacePoint (3D)
(x,y,z)
σ(x,y,z)
Direction Cosines
at (x,y,z)
σ(Dir Cosines)
dQ/dS
σ(dQ/dS)
Reference to Hits
in each view


Prong (3D)
Direction cosines at the beginning
σ(Dir cosines)
Maximum Width
Distance of MaximumWidth from start position
Collection of SpacePoints (owns)
Collection of Clusters (ref)
Vertex (3D)
Collection of Prongs (reference)
(x,y,z) where Prongs meet
Event (3D)
Collection of Vertex (reference)
Total E
σ(Total E)
Primary Vertex (reference)

1. Any object that has a collection of other objects should have that collection passed into the constructor
2. Hits should be made from unipolar pulses
3. Cluster finder algorithm should make use of DBScan/Hough line finder/Harris interest points/Roxanne's cone finding to determine final cluster product
    1. Any cluster finding algorithm needs to be able to stitch/separate collections of hits to make the final product
    2. The DBScan, etc, algorithms should be services and not modules. Making them services allows their parameters to be stored, and not being modules means you don't have to save each intermediate step
    3. Is there any need to save the intermediate information from the services?
    4. Should a cluster finder look to see if hits appear to be from overlapping sources and split them?
4. Can we make a Prong finding algorithm that takes a collection of clusters to make Prongs?
    1. We need a service to take one Cluster from each view and look for matches to make Prongs. The service would return a collection of SpacePoints for Clusters that are good matches - a Kalman Filter could be useful in this step
    2. Do we need Shower/Track objects? Can't we just use the Prong::MaximumWidth information to make that distinction? Or the dQ/dS?